Counter-Contrastive Learning for Language GANs

Yekun Chai , Haidong Zhang , Qiyue Yin and **Junge Zhang***
Institute of Automation, Chinese Academy of Sciences
chaiyekun@gmail.com haidong_zhang14@yahoo.com
{qyyin, jgzhang}@nlpr.ia.ac.cn

Abstract

Generative Adversarial Networks (GANs) have achieved great success in image synthesis, but have proven to be difficult to generate natural language. Challenges arise from the uninformative learning signals passed from the discriminator. In other words, the poor learning signals limit the learning capacity for generating languages with rich structures and semantics. In this paper, we propose to adopt the counter-contrastive learning (CCL) method to support the generator's training in language GANs. In contrast to standard GANs that adopt a simple binary classifier to discriminate whether a sample is real or fake, we employ a counter-contrastive learning signal that advances the training of language synthesizers by (1) pulling the language representations of generated and real samples together and (2) pushing apart representations of real samples to compete with the discriminator and thus prevent the discriminator from being overtrained. We evaluate our method on both synthetic and real benchmarks and yield competitive performance compared to previous GANs for adversarial sequence generation.

1 Introduction

Unsupervised text generation has achieved great success in plenty of applications, from dialogue generation to machine translation (Wu et al., 2016; Li et al., 2017). Common approaches to language models are maximizing the log-likelihood of tokens of discrete sequences given historical observations. Nevertheless, language models trained with maximum likelihood estimation (MLE) can result in exposure bias issues (Bengio et al., 2015), a distributional shift between input sequences during training and inference stages.

Generative Adversarial Networks (GANs) hold the promise of training language models, as an alternative method to MLE. GANs learn to sample during training so as to avoid the exposure bias issue, whose aim is to train a language generator to fool the discriminator that distinguishes the fake data out of real samples.

Previous innovations adopt various approaches to enhance the learning signals for generators, such as leaking information from the discriminator to the generator (Guo et al., 2017), directly matching the fake data distribution to that of real data (Zhang et al., 2017; Chen et al., 2018), learning to rank samples out of a collection of curated samples (Lin et al., 2017; Zhou et al., 2020), leveraging more powerful generator architectures to learning representations (Nie et al., 2019), *etc*. However, the problem of language GANs' training is far from being fully solved.

Inspired by the recent success in contrastive learning approaches (Chen et al., 2020) in learning effective representations, we propose a counter-contrastive learning objective to aid the adversarial learning of sequence generators in language GANs. Conventional contrastive learning methods aim at pulling positive samples together and pushing away positive samples from negative ones. In contrast, we propose counter-contrastive learning (CCL) method that (1) pulls the generated samples and real samples together (to generate real-looking data) and (2) pushes away the real samples (to hinder the training of the discriminator). Empirical results on both synthetic and real datasets demonstrate competitive results compared with previous language GANs and prove the effectiveness of our method.

2 Language GANs

Language GANs have attracted extensive interest due to their ability to mitigate the exposure bias issue. The objective of language GANs is to train a language generator $G(z;\theta^{(G)})$ that can output real-looking text samples that resemble those in the training data $p_{\text{data}}(x)$. From the game theory

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 4834–4839
November 7–11, 2021.

metaphor, language GANs consist of a generator and a discriminator playing a two-player minimax game. The generator network decodes the randomly initialized starting token z into the language sequence $G(z;\theta^{(G)})$, where the training signal is provided by the discriminator network $D(x;\phi^{(D)})$ that is trained to distinguish between the samples drawn from the real data distribution p_{data} and those produced by the generator. In this paper, we adopt the relativistic discriminator loss (Jolicoeur-Martineau, 2018) as the training objective:

$$\min_{\theta^{(G)}} \max_{\phi^{(D)}} E_{x\sim p_{\text{data}};z\sim p_z}[\log \sigma\big(D_{\phi}^{(D)}(x;\phi^{(D)}) - D_{\phi}^{(D)}(G(z;\theta^{(G)}))\big)] \quad (1)$$

where $\sigma(\cdot)$ is the sigmoid function.

There have been a large variety of language GANs that resorted to reinforcement learning (RL) heuristics with Monte Carlo search to gather the update rewards from the discriminator. The instability of RL training can further plague the reward sparsity problem. Existing work (Kusner and Hernández-Lobato, 2016; Nie et al., 2019) demonstrated that Gumbel-Softmax relaxation (Maddison et al., 2014) is effective in language GANs, thus we use the Gumbel-Softmax reparameterization instead of policy gradient in our experiments.

3 Contrastive Learning

Contrastive learning aims at learning informative representations by pulling together positive neighbors and pushing way non-neighbors (Hadsell et al., 2006). Assuming a set of paired examples $\mathcal{D} = \{(x_i, x_i^+)\}_{i=1}^N$, where (x_i, x_i^+) are positive pairs. Let the $\mathbf{h}_i$ and $\mathbf{h}_i^+$ denote the representations of x_i and x_i^+, the contrastive learning training objective is:

$$\mathcal{L}_i^{\text{CL}} = -\log \frac{e^{\text{sim}(\mathbf{h}_i,\mathbf{h}_i^+)/\tau}}{\sum_{j=1}^N e^{\text{sim}(\mathbf{h}_j,\mathbf{h}_j^+)/\tau}} \quad (2)$$

where τ is the temperature scalar, and $\text{sim}(\cdot)$ is the cosine similarity operator.

4 Methodology

4.1 Counter-Contrastive Learning

In language GANs, the discrimination classifier is prone to be overtrained, while the generator faces great challenges to obtain sufficient information for the update. To mitigate this issue, we propose a counter-contrastive learning (CCL) objective that not only renders comparative learning signals between real and fake samples but prevents the classifier from being trained too quickly.

It is crucial to construct positive and negative samples in our method. As for positive ones, we construct positive pairs by applying disparate dropout masks to get positive representations for input real texts sampled from p_{data}. Specifically, for the same real sentence, we get positive pair representations after feed them into the discriminator twice with two different random dropout operations. Denote $\mathbf{h}_i^m = f(x_i, m)$, where m is the dropout mask and f is the encoder of input sentences. In our experiments, we take the hidden representation of the last-but-one feed-forward layer in the discriminator as the representation $\mathbf{h}_i$ for each input sentence x_i.

With different dropout masks, we get the representations of positive pairs $(\mathbf{h}_i, \mathbf{h}_i^+)$. For negative samples, we randomly select fake sentences generated by the generator network and feed them into the discriminator to get fake sample representations. Therefore, we choose one from positive representations and the other from the negative to construct negative pairs $(\mathbf{h}_i, \mathbf{h}_i^-)$.

Given the mini-batch of size N, we formulate the counter-contrastive learning objectives as:

$$\mathcal{L}_i = -\log \frac{e^{\text{sim}(\mathbf{h}_i,\mathbf{h}_i^-)/\tau}}{\sum_{j=1}^N \left(e^{\text{sim}(\mathbf{h}_j,\mathbf{h}_j^-)/\tau} + e^{\text{sim}(\mathbf{h}_j,\mathbf{h}_j^+)/\tau}\right)} \quad (3)$$

where τ is the constant temperature.

Intuitively, this CCL objective aims to (1) force the fake representations to approach real data (the numerator), and (2) prevent the discriminator from learning effective representations of positive pairs by pushing away semantically close pairs (the right term in the denominator).

In contrast to contrastive learning that pulling together the positive neighbors, our CCL objective aims to draw together the fake and real samples (to let the generator imitate the real sentences) and push away the real samples (to fool and hinder the discriminator training, thereby preventing it from fast convergence).

4.2 Training Language GANs

When training the language GANs, we keep the training objective as Eq. (1) unchanged and update

the generator with Eq. (3) after the generator's conventional update.

Algorithm 1 illustrates the overall training process of the proposed framework. The discriminator and the generator could reach the Nash Equilibrium when the generator could fool the discriminator into accepting its output as being true. Since the discriminator network is easy to be overtrained, we do not pretrain it but only pretrain the generator using MLE for few epochs.

Algorithm 1 Adversarial Training of CCL.

Require: generator G_θ; discriminator D_ϕ; samples of real data $\mathcal{S}$; generator training step g; discriminator training step k; the generator pretraining epochs l.
Pretrain G_θ using MLE on $\mathcal{S}$ for l epochs
repeat
  for g steps **do**
    Sample a minibatch from real data $\mathcal{S}$
    Generate a minibatch from G_θ
    Construct **positive pairs** by feeding the real samples to D_ϕ twice with different dropout masks, and **negative samples** from $x_i^- \sim G_\theta$.
    Update G_θ via Eq. (1)
    Update G_θ via Eq. (3) (CCL training)
  end for
  for k steps **do**
    Sample a minibatch from real data $\mathcal{S}$
    Sample a minibatch from the generated data
    Train the discriminator D_ϕ by Eq. (1)
  end for
until convergence

5 Experiments

5.1 Experimental Settings

Dataset Table 1 summarizes the statistics of benchmark datasets for evaluation. We conduct experiments on both synthetic and real datasets:

- Synthetic data, which is generated by an oracle single-layer LSTM as in (Yu et al., 2017). We use a randomly initialized single-layer LSTM as the oracle, and generate 10,000 discrete sequences of length 20 and 40 respectively as either training or test set.

- Real data. We use MS COCO Image Captions (Chen et al., 2015) (only caption references are used) and EMNLP2017 WMT News dataset (Guo et al., 2017).

dataset	Synthetic data	MS COCO Image Caption	EMNLP2017 WMT News
vocabulary size	5,000	4,657	5,255
sequence length	20 / 40	37	51
training set	10,000	10,000	278,586
test set	10,000	10,000	10,000

Table 1: Summary of experimental datasets.

Evaluation Metrics For synthetic data, we use $\text{NLL}_{\text{oracle}}$ and NLL_{gen} to evaluate the quality and diversity respectively. Given the real data distribution p_{data} and fake data distribution p_θ, $\text{NLL}_{\text{oracle}}$ measures the negative log-likelihood (NLL) of generated samples $y_{1\cdots T}$ under the oracle distribution p_{data} whilst NLL_{gen} calculates the NLL of real samples $r_{1\cdots T}$ under the generated data distribution p_θ.

$$NLL_{\text{gen}} = -E_{r_{1\cdots T}\sim p_{\text{data}}} \log p_\theta(r_{1\cdots T}) \quad (4)$$

$$NLL_{\text{oracle}} = -E_{y_{1\cdots T}\sim p_\theta} \log p_{\text{data}}(y_{1\cdots T}) \quad (5)$$

For real data, it is infeasible to get an oracle to compute the $\text{NLL}_{\text{oracle}}$. We instead apply the BLEU scores (Papineni et al., 2002) to evaluate sample quality, wherein the test data serve as the reference. Besides, NLL_{gen} is adopted to evaluate the diversity of generated samples.

Baselines. Baseline models include MLE and language GANs such as SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017), LeakGAN (Guo et al., 2017), MaliGAN (Che et al., 2017), RelGAN (Nie et al., 2019), and Self-Adversarial Learning (SAL) (Zhou et al., 2020).

Model Architecture For the generator network, we apply the Relational Memory Core (Santoro et al., 2018), where the memory size is 256, the memory slot number is 1, the attention head number is 2. The input embedding dimension is set to 32. For the discriminator network, we use the multi-channel convolutional networks using filters with various window sizes to extract distinct n-gram features, followed by a max-over-time pooling operation. The input embedding dimension for the discriminator is set to 64. The filter sizes are $\{2, 3, 4, 5\}$ with the number of 300 channels for each. A max-over-time pooling and a fully connected layer is applied followed by the convolution layer.

Optimization We apply Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the initial learning rate, we set to 1e-2 and 1e-4 for pretraining and adversarial training respectively for the generator, and to 1e-4 for the discriminator during adversarial training. All trainable parameters whose L2 norm values of gradients exceed 5 are truncated.

Training Settings The following hyperparameters are finetuned: batch size of $\{32, 64, 128\}$, the CCL temperature $\tau \in \{0.2, 0.5, 1\}$. The training step for the generator and discriminator is set to $g = 1$ and $d = 5$, respectively. We pretrain the generator for 150 epochs before the adversarial training. The optimal batch size is set to 128 for both synthetic and real datasets. All experiments are conducted on Nvidia Titan RTX GPU with 5 different random seeds.

5.2 Results on Synthetic Data

Model	$\text{NLL}_{\text{oracle}}$ (20/40)	NLL_{gen} (20/40)	$\text{NLL}_{\text{oracle}}$ + NLL_{gen} (20/40)
MLE	9.05±0.03 / 9.84±0.02	5.96±0.02 / 6.55±0.02	15.02±0.03 / 16.39±0.01
SeqGAN	8.63±0.19 / 9.63±0.04	6.61±0.22 / 6.98±0.08	15.00±0.03 / 16.35±0.02
RankGAN	8.42±0.31 / 9.52±0.11	7.14±0.34 / 7.05±0.12	15.01±0.02 / 16.37±0.02
MaliGAN	8.74±0.16 / 9.67±0.03	6.62±0.25 / 7.14±0.09	15.03±0.03 / 16.39±0.03
SAL	7.71±0.17 / 9.31±0.03	6.58±0.15 / 6.97±0.05	14.29±0.11 / 16.24±0.03
Ours	**6.77**±0.34 / **6.65**±0.14	6.91±0.62 / 7.68±0.79	**13.69**±0.36 / **14.33**±0.76

Table 2: Performance of different models on the synthetic dataset with the sequence length of 20 and 40, respectively. For NLL scores, the lower, the better.

For synthetic data, we evaluate the generated sequence w.r.t. both quality and diversity. We use the oracle LSTM to evaluate the negative log-likelihood of our generated samples (denoted as $\text{NLL}_{\text{oracle}}$) to measure the quality, and the negative log-likelihood of the synthetic dataset (denoted as NLL_{gen}) measured by the generator during training. We also report the best $\text{NLL}_{\text{oracle}}$+$\text{NLL}_{\text{gen}}$ to evaluate the trade-off between quality and diversity. It is observed that our model outperforms baseline models in terms of quality (measured by $\text{NLL}_{\text{oracle}}$) and quality-diversity trade-off (measured by $\text{NLL}_{\text{oracle}}$+$\text{NLL}_{\text{gen}}$), and achieves or matches the competitive results of baselines w.r.t. the diversity (indicated by NLL_{gen}).

5.3 Results on Real Data

Table 3 exhibits the final results of the BLEU and NLL_{gen} scores on different comparison models. Notably, our model shows a significant improvement over previous methods, consistently achieves competitive results in terms of the sample quality (indicated by BLEU scores) while maintaining the diversity (indicated by NLL_{gen}). Table 4 shows the same trend on EMNLP2017 WMT News dataset.

Model	BLEU-2	BLEU-3	BLEU-4	BLEU-5	NLL_{gen}
MLE	0.731	0.497	0.305	0.189	0.718
SeqGAN	0.745	0.498	0.294	0.180	1.082
RankGAN	0.743	0.467	0.264	0.156	1.344
LeakGAN	0.746	0.528	0.355	0.230	0.679
RelGAN	0.849±0.030	0.687±0.047	0.502±0.048	0.331±0.044	0.756±0.054
SAL	0.785±0.02	0.581±0.03	0.362±0.02	0.227±0.02	0.873±0.02
Ours (CCL)	**0.871**±0.032	**0.715**±0.050	**0.538**±0.068	**0.399**±0.082	**0.630**±0.103

Table 3: BLEU and NLL_{gen} on MS COCO image captions. For BLEU scores, the higher, the better.

Model	BLEU-2	BLEU-3	BLEU-4	BLEU-5	NLL_{gen}
MLE	0.768	0.473	0.240	0.126	2.382
SeqGAN	0.777	0.491	0.261	0.138	2.773
RankGAN	0.727	0.435	0.209	0.101	3.345
LeakGAN	0.826	0.645	0.437	0.272	**2.356**
RelGAN	0.881±0.013	0.705±0.019	0.501±0.023	0.319±0.018	2.482±0.031
SAL	0.788±0.02	0.523±0.02	0.281±0.02	0.149±0.02	2.578±0.04
Ours	**0.903**±0.016	**0.749**±0.022	**0.525**±0.017	**0.324**±0.008	2.818±0.499

Table 4: BLEU and NLL_{gen} on EMNLP2017 WMT News dataset.

5.4 Analysis

Ablation Test To further verify the benefits of our method, we conduct an ablation test by removing the CCL update on MS COCO image captions. It can be seen from Table 5 that ablating the CCL component can quantitatively decrease the model performance: the sentence quality decreased (with the decrease of BLEU scores) and the diversity drops (with the increase of NLL_{gen} metric).

Model	BLEU-2	BLEU-3	BLEU-4	BLEU-5	NLL_{gen}
Ours	0.872	0.715	0.531	0.363	0.610
w/o CCL	0.813⇓	0.630⇓	0.445⇓	0.312⇓	0.683⇑

Table 5: Ablation test. The performance drops after ablating the CCL method.

Comparison between Generated Samples For fair comparison, we select the generated sentences that contain the word "cat" from samples produced by models with and without the CCL method (see Table 6). It is observed that GANs with CCL tend to produce sentences with better diversity. For example, with the structure "a cat is sitting on top of a car", models w/ CCL can enrich it with different modifier words. However, after removing CCL, the model can duplicate words such as "sitting" regardless of its repetitive usage. Moreover, as shown in the last row of Table 6, with the CCL method, the

language GANs tend to write semantically meaningful samples in comparison with the counterpart without CCL.

model	Sample sentences
w/o CCL	a cat is sitting on a white plate .
	a cat is sitting on a bathroom sink sitting inside of a toilet .
	a black and white cat outside decorated in rustic kitchen .
	a cat is sitting on a bathroom sink sitting in a bathroom .
	a cat is sitting on a bathroom sink sitting on a bathroom counter .
	a cat sitting on a gravel ground inside of a bathroom sink .
	a cat is sitting on a bathroom sink sitting in a bathroom .
w/ CCL	a cat is sitting on top of a car .
	a cat is sitting on top of a car cleaning itself .
	a cat is sitting on top of a car roof .
	a cat is sitting on top of a car hood .
	a cat is sitting on top of a man 's head in front of a glass door .
	a dog sitting on top of a parked car near a cat .
	a cat in a white bathroom with a toilet paper beside a child .

Table 6: Comparison between generated sentences from models with and without counter-contrastive learning approach.

6 Related Work

A variety of language GANs integrated the RL paradigm into GANs. SeqGAN (Yu et al., 2017) firstly takes the text generation as a Markov decision-making process and trains the language generator with the policy gradient algorithm. RankGAN (Lin et al., 2017) and SAL (Zhou et al., 2020) enrich the restrictive signals by ranking constructed pairs. LeakGAN (Guo et al., 2017) leaks the hidden states of the generator to promote the generator training.

Another line of previous work either approximates the categorical sampling or optimizes on continuous representations, such as Gumbel-Softmax GAN (Kusner and Hernández-Lobato, 2016), TextGAN (Zhang et al., 2017), FMGAN (Chen et al., 2018) and RelGAN (Nie et al., 2019).

Our work aims to integrate the prevalent contrastive learning approach in supporting the generator training, which lies in the line of methods using comparative signals or ranking classifiers, such as RankGAN and SAL. From the perspective of feature matching, the counter-contrastive learning objective can be considered as a contrastive signal to draw together the fake and real sample representations.

7 Conclusion

In this paper, we introduce a counter-contrastive learning objective to advance the training of language GANs. It pulls the representation of generated and real samples together to promote the generator training, and pushes apart real sample pairs to depress the discriminator training as a competitor. Our future work will include extending the counter-contrastive learning method to other text generation tasks such as machine translation and dialogue generation.

Acknowledgements

The authors thank all anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (Grant No.61876181), Beijing Nova Program of Science and Technology under Grant No.Z191100001119043, and in part by the Youth Innovation Promotion Association, CAS.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Tong Che, Yanran Li, R. Zhang, R. Devon Hjelm, W. Li, Y. Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *ArXiv*, abs/1702.07983.

Liqun Chen, Shuyang Dai, Chenyang Tao, Dinghan Shen, Zhe Gan, H. Zhang, Yizhe Zhang, and L. Carin. 2018. Adversarial text generation via feature-mover's distance. In *NeurIPS*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709.

Xinlei Chen, H. Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. L. Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *ArXiv*, abs/1504.00325.

Jiaxian Guo, S. Lu, Han Cai, W. Zhang, Y. Yu, and J. Wang. 2017. Long text generation via adversarial training with leaked information. In *AAAI*.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2:1735–1742.

Alexia Jolicoeur-Martineau. 2018. The relativistic discriminator: a key element missing from standard gan. In *ICLR*.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution. *ArXiv*, abs/1611.04051.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

Kevin Lin, Dianqi Li, Xiaodong He, M. Sun, and Zhengyou Zhang. 2017. Adversarial ranking for language generation. In *NIPS*.

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *NIPS*.

Weili Nie, Nina Narodytska, and Ankit B. Patel. 2019. Relgan: Relational generative adversarial networks for text generation. In *ICLR*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. 2018. Relational recurrent neural networks. In *NeurIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Lantao Yu, W. Zhang, J. Wang, and Y. Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Yizhe Zhang, Zhe Gan, K. Fan, Z. Chen, Ricardo Henao, Dinghan Shen, and L. Carin. 2017. Adversarial feature matching for text generation. In *ICML*.

Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and M. Zhou. 2020. Self-adversarial learning with comparative discrimination for text generation. In *ICLR*.